Frontend technologies used in our case

2023-11-14 version by JV

*SPA = **Single-Page Application**, Only one web page downloaded from the Web Server, but then with JS & AJAX that single page's DOM updated constantly. Showing/hiding certain Views so that it looks like we would have several Pages*

React: SPA component library

- renders the output page (HTML+CSS+JS) DOM based on 1) your Page template code (React Components inside other React components) AND 2) the data provided by AJAX

React routing: SPA front-end routing between the Views (~like "going" to different page)

- though really just showing and hiding React Views (React components meant to be navigable pages) in the DOM.
- we can also pass parameter data while navigating to another View, e.g. id:s

React-Material-UI: Material-UI styled React components with common theme and styles

- Those React components need to also be 1) React-routed and 2) Redux-connected!

Other libs, luxon (dates and times), papaparse (JSON<->CSV parsing), free font and icon libs, react-progress-bar (result view bars), formik (React forms), … axios (AJAX, though replaced by standard fetch)

Dev time tools, no need to memorize: vite, rimraf, tsc/typescript, selenium, biome, pre-commit, npm, nano-staged,

Redux: Front-end side (=in browser memory) State management.

- part of the data model temporarily kept in browser memory (in Redux store/state)
- update frequency depends on the nature of the data (colors maybe not before next login, messages maybe polled/refreshed a lot more often)
- many components can share the same data

React-Redux: React components will be data- and event-bound to Redux store.

- actions dispatched to Redux code.
- redux store state bound to the React state of the components